**SENTIMENT ANALYSER**

Now we come to our project, Sentiment Analyzer. It is going to be detecting your current mood. For example, if you had a very stressful day, it will ask you a certain set of questions and accordingly it will analyze your answers, we are running some conditional statements and through this statement we are polarizing your sentiments.

By this, what we mean is we are setting some set of questions and at some point, of time it will eventually match and accordingly we have a set of instructions assigned to them which are going to be executed for example if you are having a bad mood then we can probably redirect you to some humorous or funny videos which will eventually uplift your mood which can make you feel much better. If you are just tired after a hectic day then we might order you some food which will include some of your favorite cuisines from your favorite restaurant. It will directly redirect you to the payment portal where by just tapping the pay option you get your food delivered at your doorstep. While if you are in a relaxed state of mind, we can put some music on that might make you feel even more better. By this process we can analyze the sentiments of a person and hence we named as a Sentiment Analyzer.

In project Sentiment Analyzer, we used Python 3.8. On this platform we used several modules to make our code more efficient and user friendly. The modules are as follows:

1. TextBlob- TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

TextBlob stands on the giant shoulders of [NLTK](http://www.nltk.org/) and [pattern](http://www.clips.ua.ac.be/pages/pattern-en), and plays nicely with both.

## **Features:**

* Noun phrase extraction
* Part-of-speech tagging
* Sentiment analysis
* Classification (Naive Bayes, Decision Tree)
* Tokenization (splitting text into words and sentences)
* Word and phrase frequencies
* Parsing
* **n**-grams
* Word inflection (pluralization and singularization) and lemmatization
* Spelling correction
* Add new models or languages through extensions
* WordNet integration

1. Webbrowser- In Python, **webbrowser module** provides a high-level interface which allows displaying Web-based documents to users. The webbrowser module can be used to launch a browser in a platform-independent manner.

The [webbrowser](https://docs.python.org/3/library/webbrowser.html" \l "module-webbrowser" \o "webbrowser: Easy-to-use controller for Web browsers.) module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the [open()](https://docs.python.org/3/library/webbrowser.html#webbrowser.open) function from this module will do the right thing.

Under Unix, graphical browsers are preferred under X11, but text-mode browsers will be used if graphical browsers are not available or an X11 display isn’t available. If text-mode browsers are used, the calling process will block until the user exits the browser.

If the environment variable BROWSER exists, it is interpreted as the [os.pathsep](https://docs.python.org/3/library/os.html" \l "os.pathsep" \o "os.pathsep)-separated list of browsers to try ahead of the platform defaults. When the value of a list part contains the string %s, then it is interpreted as a literal browser command line to be used with the argument URL substituted for %s; if the part does not contain %s, it is simply interpreted as the name of the browser to launch.

1. Time- This module provides various time-related functions. For related functionality, see also the [datetime](https://docs.python.org/3/library/datetime.html#module-datetime) and [calendar](https://docs.python.org/3/library/calendar.html#module-calendar) modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

* The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).
* The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding [leap seconds](https://en.wikipedia.org/wiki/Leap_second). Leap seconds are excluded from this total on all POSIX-compliant platforms.
* The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.
* Function [strptime()](https://docs.python.org/3/library/time.html#time.strptime) can parse 2-digit years when given %y format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.
* UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.
* DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year. DST rules are magic (determined by local law) and can change from year to year. The C library has a table containing the local rules (often it is read from a system file for flexibility) and is the only source of True Wisdom in this respect.
* The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g., on most Unix systems, the clock “ticks” only 50 or 100 times a second.